Anlage 10: Anleitung Experimentierplatte

Wie in der untenstehenden Abbildung zu sehen, ist die Experimentierplatte „Controllino MAXI Automation“ in fünf Teile gegliedert. Auf den folgenden Seiten wird der Aufbau und die Nutzung der Experimentierplatte beschrieben.

1. Kompakt-SPS „Controllino MAXI Automation“;
2. Arduino LCD-Shield;
3. Analoge und digitale Eingänge;
4. Analoge und digitale Ausgänge;
5. Spannungsversorgung

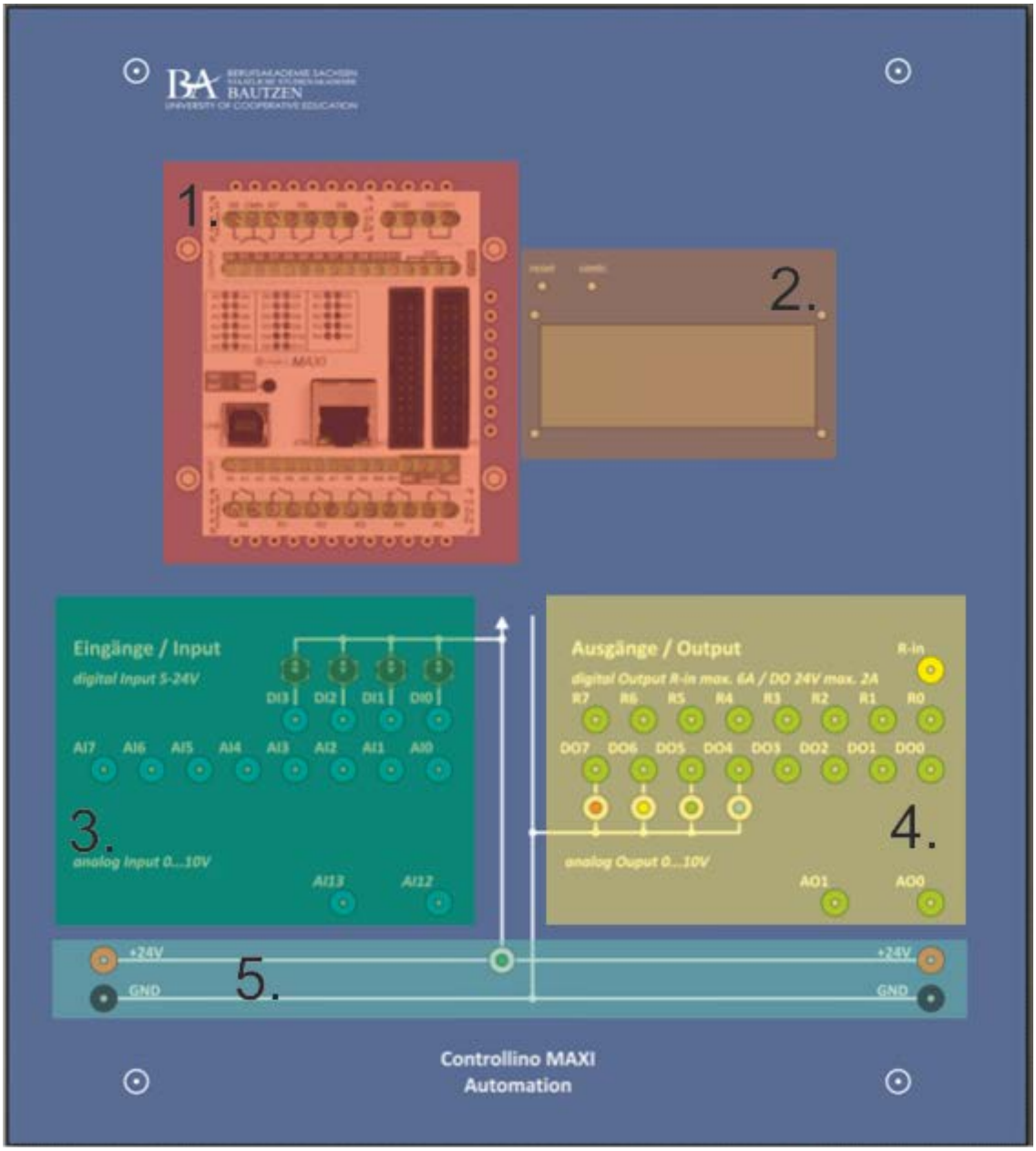

Abb. 1: Einteilung der Experimentierplatte (Quelle: Prof. Leander Mirke)

Aufbau

Der erste Teil der Experimentierplatte ist mit der Kompakt-SPS „Controllino MAXI Automation“ besetzt. In der folgenden Tabelle sind alle Ein- und Ausgänge aufgelistet, welche nach außen auf die Anschlussbuchsen bzw. die Leiterplatte (Blau markiert) geführt werden. Des Weiteren sind alle technische Daten zu den Ein- und Ausgängen verzeichnet, welche vom Nutzer beachtet werden sollten. Die digitalen und A/D-Eingänge werden durch Optokoppler galvanisch von der SPS getrennt.

Tab. 1: Ein- und Ausgänge (Quelle: eigene Darstellung)

MAXI Automation				
		Anzahl	Genutzt	Technische Daten
Ein- und Ausgänge	Analoge Eingänge (0-10V)	2	2	0...10V
	Analoge Ausgänge (0-10V)	2	2	0...10V oder 0...20mA
	A/D-Eingänge (0-24V)	12	8	0V - 26,4V
	Digitale Eingänge (24V)	4	4	low Level 0V - 7,2V / high Level 18V - 26,4V
	Interrupteingänge (24V)	2	0	
	Digitale Ausgänge (24V) davon PWM-fähig	8	8	low Level 0V - 4,8V / high Level Vin - 10%
		8		15% - 85%
	Relaisausgänge (bis 230V)	10	8	230V 6A AC /30V 6A DC

Außerdem stehen dem Nutzer noch vier Kippschalter zur Verfügung, welche direkt 24V auf die vier digitalen Eingänge schalten können. Um etwaige Betriebszustände anzeigen zu können, wurden im vierten Teil vier LED an den digitalen Ausgängen installiert. Des Weiteren wird durch eine LED im fünften Teil angezeigt ob die

Experimentierplatte mit Spannung versorgt wird. Die Kommunikation zwischen der Kompakt-SPS und dem LCD-Shield wird über die SPI-Schnittstelle der SPS realisiert. In Abbildung 2 ist die dazugehörige Schaltung dargestellt.

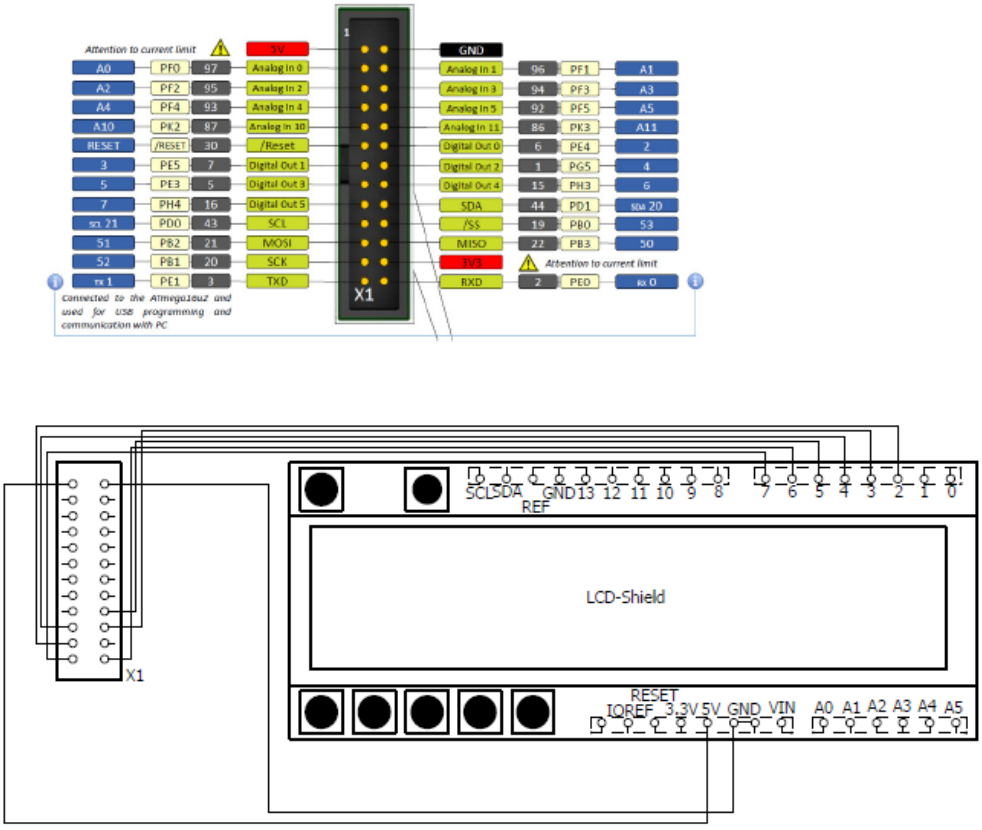

Abb. 2: Stromlaufplan Anschluss LCD-Shield (Quelle: eigene Darstellung)

Nutzung

Über die USB-Schnittstelle der Kompakt-SPS „Controllino MAXI Automation“ kann das in Arduino IDE erstellte Programm hochgeladen werden. Dazu müssen entsprechende Voreinstellungen in der Entwicklungsumgebung getroffen werden (siehe Bedienungsanleitung Controllino).

In der Abbildung 3 zusehenden Menüleiste befinden sich die wichtigsten Elemente der Entwicklungsumgebung, beispielsweise die Syntaxüberprüfung oder das Hochladen des Sketches (Skizze), welcher den Programmcode enthält.

Im Wesentlichen besteht das Programm aus der anfänglichen Variablendeklaration und zwei Hauptfunktionen. Der Setup-Routine (void setup), welche Anweisungen enthält die einmalig zum Programmstart ausgeführt werden und die Loop-Funktion (void loop), in der sich der eigentliche Programmcode befindet und die als Endlosschleife ausgeführt wird. In Abbildung 3 wird über den Controller „Controllino MAXI Automation“ ein LCD-Shield angesteuert werden, welches den Text „Hallo Welt!“ anzeigen soll.

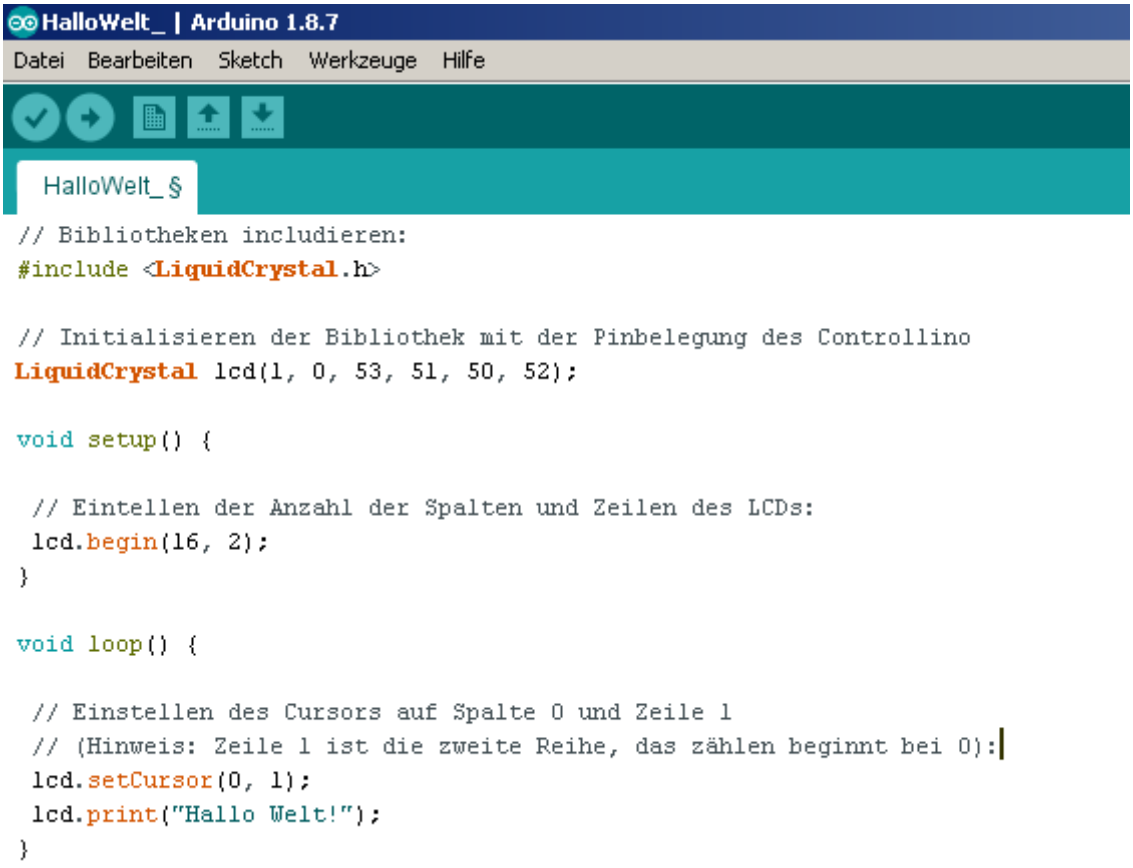

Abb. 3: Programmierbeispiel "Hallo Welt!" (Quelle: eigene Darstellung)

Um das LCD-Shield ansteuern zu können, muss in zweiten Schritt die entsprechende Bibliothek <LiquidCrystal.h> inkludiert werden. Als Nächstes folgt die Initialisierung der inkludierten Bibliothek entsprechend der Pin-Belegung am „Controllino“. Wie in Abbildung 3 zu sehen, wird in der ersten Hauptfunktion (void setup) die Einstellung der Anzahl der Zeilen und Spalten des LCD-Shields vorgenommen. Anschließend wird in der zweiten Hauptfunktion (void loop), welche den eigentlichen Programmcode enthält, der Cursor in die gewünschte Zeile und Spalte gesetzt. Diese Zeile und Spalte markiert den entsprechenden Beginn des Textes auf dem LCD-Shield. Im letzten Schritt der Programmierung wird über dem print-Befehl der Text definiert der angezeigt

werden soll. Abschließend wird der Programmcode kompiliert und auf den Controller hochgeladen.

In den folgenden Abbildungen werden Code- und Anschlussbeispiele für die Nutzung:

1. der analogen und digitalen Eingänge,
2. der analogen und digitalen Ausgänge,
3. der PWM-Ausgänge mit Servomotor,
4. der Relais und
5. dem Moduswechsel dargestellt.

Analoge und digitale Eingänge

```
5  int Poti = CONTROLLINO_AI12;                    // Sollwertvorgabe Servomotor durch Poti
6  int Opto = CONTROLLINO_AI13;                    // Istwertmessung Gabellichtschranke
7  int value;                                      // erstellen der Variable value
8  int servo = CONTROLLINO_D4;                     // Sollwertgeber Servo
9
10 // Initialisierung der LC-Bibliothek mit Angabe der Pins vom Controllino MAXI Automation
11
12 LiquidCrystal lcd(1 , 0, 53, 51, 50, 52);
13
14 // Objekt servo1 erstellen
15
16 Servo servo1;
17
18 void setup()
19 {
20               servo1.attach(servo, 0, 180);      // Zuweisung PWM Ausgang D4 (servo) zu Servo-Objekt
21               lcd.begin(16, 2);                  // Angabe Zeilen und Spalten LCD
22               lcd.setCursor(0, 0);               // setze Cursor zum nullten Zeichen in Zeile 0
23 }
24
25 void loop()
26 {
27               value = analogRead(Poti);
28               value = map(value, 0, 1023, 180, 0)  // Skalierung Int-Wertes Poti auf Stellbereich Servo
29               servo1.write (value);              // Int-Wert des Poti auf Servomotor übertragen
30               lcd.setCursor(1, 0);
31               lcd.print("Automatik");
32               lcd.setCursor(1, 1);
33               lcd.print("Soll:  ");
34               lcd.print(value);                  // Anzeige: "Soll: Sollwert Poti"
35 }
```

Abb. 3: Analoge Eingänge (Quelle: eigene Darstellung)

```
#include <Controllino.h>

void setup()
{
  pinMode(CONTROLLINO_D0, INPUT);        // D0 als Input
  pinMode(CONTROLLINO_D1, INPUT);        // D1 als Input
}

void loop()
{
  T0 = digitalRead(Taster0);             // Eingang Taster0 lesen
  T1 = digitalRead(Taster1);             // Eingang Taster1 lesen

  if (T0 == HIGH) mode = 1;              // Bei Druck des Taster 0 -> Unterprogramm "Schrittbetrieb"
  if (T1 == HIGH) mode = 2;              // Bei Druck des Taster 1 -> Unterprogramm "Automatik"

}
```

Abb. 4: Digitale Eingänge (Quelle: eigene Darstellung)

Analoge und digitale Ausgänge

```
#include <Controllino.h>

void setup()
{

 pinMode(CONTROLLINO_AO0, OUTPUT);
 pinMode(CONTROLLINO_AO1, OUTPUT);
}

void loop()
{
 int analogOut0 = 127;                    // 0 - 255 einstellbar (0 - 10V oder 0 - 20mA)
 int analogOut1 = 255;                    // 0 - 255 einstellbar (0 - 10V oder 0 - 20mA)
 analogWrite(CONTROLLINO_AO0, analogOut0); // Stellen des Analogausgangs auf 0 bis 5 V oder 10 mA
 analogWrite(CONTROLLINO_AO1, analogOut1); // Stellen des Analogausgangs auf 1 bis 10 V oder 20 mA
}
```

Abb. 5: Analoge Ausgänge (Quelle: eigene Darstellung)

```
#include <Controllino.h>
#include <LiquidCrystal.h>

void setup()
{
 pinMode(CONTROLLINO_D7, OUTPUT);
 pinMode(CONTROLLINO_D6, OUTPUT);
  pinMode(CONTROLLINO_D5, OUTPUT);
 pinMode(CONTROLLINO_D4, OUTPUT);
}

void loop()
{
 delay(2000);
 digitalWrite(CONTROLLINO_D7, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D7, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_D6, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D6, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_D5, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D5, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_D4, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D4, LOW);
 delay(100);
}
```

Abb. 6: Digitale Ausgänge (Quelle: eigene Darstellung)

PWM-Ausgänge mit Servomotor

```
#include <Controllino.h>
#include <Servo.h>

int Poti = CONTROLLINO_AI12;              // Sollwertvorgabe Servomotor durch Poti
int value


void setup()
{
  servo.attach(CONTROLLINO_D4);           // Zuweisung PWM Ausgang D4 zu Servo-Objekt
}

void loop()
{
    value = analogRead(Poti);             // lesen des Analogwertes des Poti und schreiben auf die Variable "Poti"
    value = map(value, 0, 1023, 180, 0);  // Skalierung des Int-Wertes des Poti auf Stellbereich Servo
    servo.write (value);                  // Int-Wert des Poti auf Servomotor übertragen
}
```

Abb. 7: PWM-Ausgänge mit Servomotor (Quelle: eigene Darstellung)

Relais

```
#include <Controllino.h>

void setup()
{
 pinMode(CONTROLLINO_R0, OUTPUT);     // R0 als Ausgang setzen
}


void loop()
{
 digitalWrite(CONTROLLINO_R0, HIGH); // Relais auf High setzen
 delay(100);                         // 100ms warten
 digitalWrite(CONTROLLINO_R0, LOW);  // Relais auf Low setzen
 delay(100);                         // 100ms warten
}
```

Abb. 8: Relais (Quelle: eigene Darstellung)

Moduswechsel

```
#include <Controllino.h>

int LED1 = CONTROLLINO_D7;                  // LED rot
int LED2 = CONTROLLINO_D6;                  // LED gelb
int Taster0 = CONTROLLINO_DI0;               // Taster DI0
int Taster1 = CONTROLLINO_DI1;               // Taster DI1
int mode;                                   // erstellen der Variable mode (Modus)
int T0 = 0;                                 // Variable zum Lesen des Status des Taster0
int T1 = 0;                                 // Variable zum Lesen des Status des Taster1

void setup()
{
  pinMode(Taster0, INPUT);                          // Taster0 als Input
  pinMode(Taster1, INPUT);                          // Taster1 als Input
  pinMode(LED1, OUTPUT);                            // LED1 als Output
  pinMode(LED2, OUTPUT);                            // LED2 als Output
}

void loop()
{
  if (digitalRead(Taster0) == HIGH)LED1zeigen();
  if (digitalRead(Taster1) == HIGH)LED2zeigen();
}
void LED1zeigen()
{
                  digitalWrite(LED1, HIGH);
}
void LED2zeigen()
{
                  digitalWrite(LED2, HIGH);
}
```

Abb. 9: Moduswechsel (Quelle: eigene Darstellung)